EDB™ Sustainable Performance Profiling in PostgreSQL

Dirk Krautschick

PGConf.DE 2023

#whoami

Dirk Krautschick

Sales Engineer

- With EDB since 09/2022
- 15 years DBA, Trainer & Consulting
- PostgreSQL, Oracle

- Married, 2 Junior DBAs
- Mountainbike, swimming, movies, music, hifi/home cinema, 8bit computing




@d33_k4y


EDB

Disclaimer

Different audience, different perspectives

My experience, my honest opinion

Let‘s stay open minded

Always open for discussions

What happened so far...

What happened so far…

There was a talk….

"Pro-Active Performance Analysis in PostgreSQL"

https://www.youtube.com/watch?v=rgdA0FwVShI

About

- Performance problems overall
- Different analysis approaches
- Recommendations/usage of Extensions

EDB

What happened so far...

There was a talk....

Solid feedback

Consensus in practical experiences

Many questions about “THAT LAST PART of the talk”

Motivation to do a Spin-Off talk :-)

The Idea

The Idea

Information sources for Performance analytics

- Obvious Sources
  - Parameters, Sizing (at that time!)
  - Information_schema, system catalogues
  - Logging
  - `PG_STAT_STATEMENTS` (Extension)

The Idea

Information sources for Performance analytics

More special Sources

- `PG_STAT_KCACHE` (Extension)
- `PG_WAIT_SAMPLING` (Extension)
- …

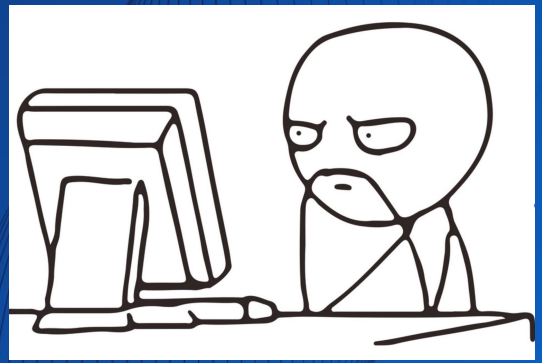
The Idea

Getting sustainable?

Ring-Buffer-like settings in extensions are volatile

Several different retention

Several Views, Tables…but also volatile



How to handle the collection of all that information?

The Idea

- putting all in a repository/database
- while handling the retention of all information

EDB

The Idea

What about logging …

- PostgreSQL logging is awesome
- Exhaustive possibilities
- Straight and easy configuration
- Be aware of storage and load
  - High maintenance
  - Evaluate Logging strategies

```
log_line_prefix = '%t [%p]:
user=%u,db=%d,app=%a,client=%h ,
…
log_parser_stats = off
log_planner_stats = off
log_executor_stats = off
log_statement_stats = on
…
log_checkpoints = on
log_connections = on
log_disconnections = on
log_lock_waits = on
log_temp_files = 0
log_autovacuum_min_duration = 0
log_error_verbosity = default
…
log_min_messages = debug5
log_min_error_statement = debug5
log_min_duration_statement = 0
log_min_duration_sample = 0
…
log_statement = 'all'
```

The Idea

… and what about PGBadger?

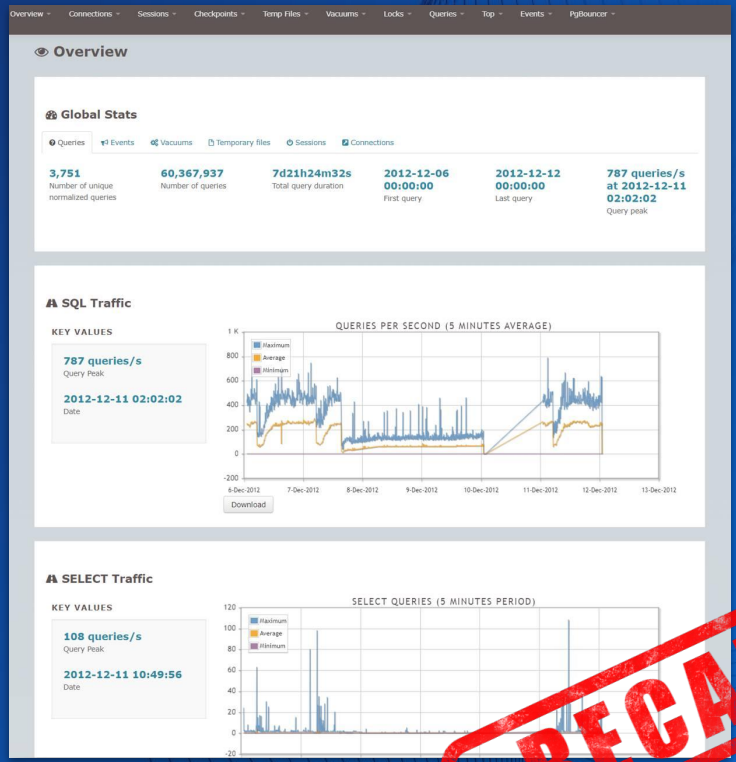
Advanced logging analysis reporting

https://pgbadger.darold.net/

Incremental daily/Cumulative weekly reports

The right direction, but

- Massive logging necessary
- Log file handling



RECAP

The Idea

And what about Monitoring …

For sure, monitoring is essential!

„The fact that it‘s slow or what is exactly slow?“

Necessary PostgreSQL insights necessary

Pointing out a problem

EDB

Think outside the Box

Think outside the Box

What about the other Database Systems?

Let’s pick a random example…

… like … Oracle Database :-)

Collects almost everything per default (sometimes sampled)

Interpretation with

- Querying Views (obviously!)
- Statspack (basic, always available and “costless”)
- Diagnostic and Tuning Pack (expensive Option)
  - Only for Enterprise Edition
  - Several Tools, like ASH, AWR,...

EDB

Think outside the Box

The Oracle Approach

Frequent snapshots of Performance Data in a Repository

Defined time periods and retention

Creation of reports based on those snapshots

- Time frame between two or more snapshots

Same Idea Statspack and AWR

Think outside the Box

Oracle Automatic Workload Repository (AWR)

WORKLOAD REPOSITORY PDB report (root snapshots)

DB Name	DB Id	Unique Name	Role	Edition	Release	RAC	CDB
CDB1P	281979110	CDB1P_fra1hv	PRIMARY	XP	19.0.0.0.0	YES	YES

Instance	Inst Num	Startup Time	User Name	System Data Visible
CDB1P1	1	21-Jan-23 23:54	SYS	YES

Container DB Id	Container Name	Open Time
1186506325	PDB1	21-Jan-23 23:55

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
exaccnode1	Linux x86 64-bit	8	4	2	125.41

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	13678	16-Feb-23 08:41:13	74	8.3
End Snap:	13679	16-Feb-23 09:40:40	72	8.4
Elapsed:		59.45 (mins)		
DB Time:		79.94 (mins)		

Report Summary

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1.3	0.1	0.00	0.02
DB CPU(s):	0.7	0.1	0.00	0.01
Background CPU(s):	0.0	0.0	0.00	0.00
Redo size (bytes):	733,057.3	57,150.8		
Logical read (blocks):	130,864.4	10,202.5		
Block changes:	2,818.6	219.8		
Physical read (blocks):	2,374.9	185.2		
Physical write (blocks):	0.0	0.0		
Read IO requests:	258.3	20.1		
Write IO requests:	0.0	0.0		
Read IO (MB):	18.6	1.5		
Write IO (MB):	0.0	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
Global Cache blocks received:	134.9	10.5		
Global Cache blocks served:	0.0	0.0		
User calls:	54.3	4.2		
Parses (SQL):	70.3	5.5		
Hard parses (SQL):	0.5	0.0		
SQL Work Area (MB):	2.5	0.2		
Logons:	0.5	0.0		
User logons:	0.0	0.0		
Executes (SQL):	473.4	36.9		
Rollbacks:	0.5	0.0		
Transactions:	12.8			

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
DB CPU		2361,7		49.2	
buffer busy waits	1,060,499	1142,4	1.08ms	23.8	Concurrency
SQL*Net message from dblink	7,053	308,3	43.71ms	6.4	Network
resmgr:cpu quantum	30,567	178,4	5.84ms	3.7	Scheduler
cell smart table scan	70,995	35	492.69us	.7	User I/O
SQL*Net more data from dblink	38,341	34,5	899.55us	.7	Network
JS kgl get object wait	345	31,7	91.82ms	.7	Administrative
gc current block 3-way	106,559	29,4	276.10us	.6	Cluster
cell multiblock physical read	25,288	26,6	1.05ms	.6	User I/O
cell single block physical read: RDMA	466,602	24,2	51.95us	.5	Other

Wait Classes by Total Wait Time

Wait Class	Waits	Total Wait Time (sec)	Avg Wait Time	% DB time	Avg Active Sessions
DB CPU		2,362		49.2	0.7
Concurrency	1,125,664	1,185	1.05ms	24.7	0.3
Network	187,985	345	1.83ms	7.2	0.1
Scheduler	30,574	178	5.84ms	3.7	0.1
Cluster	599,690	153	254.69us	3.2	0.0
Other	671,133	89	132.03us	1.8	0.0
User I/O	121,279	69	568.13us	1.4	0.0
Administrative	345	32	91.82ms	.7	0.0
Application	52,545	25	474.13us	.5	0.0
Commit	16,422	11	669.09us	.2	0.0
System I/O	6,981	1	138.07us	.0	0.0
Configuration	618	0	532.75us	.0	0.0

IO Profile

	Read+Write Per Second	Read per Second	Write Per Second
Total Requests:	259.2	259.2	0.0
Database Requests:	258.3	258.3	0.0
Optimized Requests:	259.2	259.2	0.0
Redo Requests:			
Total (MB):	18.6	18.6	0.0
Database (MB):	18.6	18.6	0.0
Optimized Total (MB):	18.6	18.6	0.0
Redo (MB):			
Database (blocks):	2,374.9	2,374.9	0.0
Via Buffer Cache (blocks):	278.2	278.2	0.0
Direct (blocks):	2,096.7	2,096.7	0.0

Database Resource Limits

	Begin	End
CPUs:	8	8
SGA Target:	17,179,869,184	17,179,869,184
PGA Target:	6,442,450,944	6,442,450,944
Memory Target	0	0

Main Report

- Report Summary
- Wait Events Statistics
- SQL Statistics
- Instance Activity Statistics
- IO Stats
- Wait Statistics
- Undo Statistics
- Segment Statistics
- In-Memory Segment Statistics
- Dictionary Cache Statistics
- Library Cache Statistics
- Initialization Parameters
- Active Session History (ASH) Report
- ADDM Reports

Exadata Configuration and Statistics

- Exadata Report Summary
- Exadata Server Configuration
- Exadata Server Health Report
- Exadata Statistics

Back to Top

Wait Events Statistics

- Time Model Statistics
- Foreground Wait Class
- Foreground Wait Events
- Service Statistics
- Service Wait Class Stats
- Top Process Types by Wait Class
- Top Process Types by CPU Used

Back to Top

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 43.3% of Total DB Time (s): 4,796
- Captured PL/SQL account for 94.4% of Total DB Time (s): 4,796

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	PDB Name	SQL Text
3,296.05	13,662	0.24	68.72	47.10	0.01	747d8puab2h4u	OEM.CacheModeWaitPingPool	PDB1	BEGIN DBMS_APPLICATION_INFO.SE...
1,320.15	50,504	0.03	27.52	72.83	0.00	65x06n09f0v45	OEM.CacheModeWaitPingPool	PDB1	select /*+ first_rows(1) no_ex...
228.18	1	228.18	4.76	3.65	0.01	d2kggafaxdxp6	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
225.80	1	225.80	4.71	2.72	0.00	6pw18wny03n9q	DBMS_SCHEDULER	PDB1	/* MV_REFRESH (INS) */INSERT /...
178.12	3,735	0.05	3.71	64.93	0.44	c9171b67uapwk	OEM.PbsSystemPool	PDB1	BEGIN em_metric.bulk_insert_me...
153.16	60	2.55	3.19	86.61	0.24	1t83y2u960dxd	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
100.08	1	100.08	2.09	41.85	8.30	c23jwp2pztmbs	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
96.36	1	96.36	2.01	19.11	0.89	ayjdzzm9ytydr	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
82.35	5	16.47	1.72	93.25	0.00	7y0xytx71ss2g	MGMT_COLLECTION.COLLECTION_SUBSYSTEM	PDB1	CALL emd_rac.service_status_bu...
76.25	1,293	0.06	1.59	93.63	0.00	gk1qggpwcaugv	MGMT_COLLECTION.COLLECTION_SUBSYSTEM	PDB1	WITH CLS_INFO AS (SELECT DEST...
69.15	3,660	0.02	1.44	68.42	0.57	btf5nuff4cm3a	OEM.PbsSystemPool	PDB1	INSERT INTO EM_METRIC_VALUES (...
63.75	1	63.75	1.33	84.16	4.60	46z58hrjxrh4x	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
56.98	1	56.98	1.19	25.49	1.32	c3mqytm5242h1	DBMS_SCHEDULER	PDB1	/* MV_REFRESH (INS) */INSERT /...
54.82	620	0.09	1.14	44.93	0.53	b12g3n0660rnc	OEM.PbsDefaultPool	PDB1	BEGIN EMD_LOADER.EXECUTE_POST_...
54.14	620	0.09	1.13	45.02	0.54	7x0yk3yrzv3rf	OEM.PbsDefaultPool	PDB1	BEGIN ECM_CT.POSTLOAD_CALLBACK...
49.22	179	0.27	1.03	20.54	46.58	bjdntqgd0fny4	OEM.PbsDeprecatedJobWorkerPool	PDB1	SELECT context_type AS name, t...

Back to SQL Statistics
Back to Top

SQL ordered by CPU Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- %Total - CPU Time as a percentage of Total DB CPU
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 54.3% of Total CPU Time (s): 2,362
- Captured PL/SQL account for 93.7% of Total CPU Time (s): 2,362

CPU Time (s)	Executions	CPU per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	PDB Name	SQL Text
1,552.48	13,662	0.11	65.74	3,296.05	47.10	0.01	747d8puab2h4u	OEM.CacheModeWaitPingPool	PDB1	BEGIN DBMS_APPLICATION_INFO.SE...
961.47	50,504	0.02	40.71	1,320.15	72.83	0.00	65x06n09f0v45	OEM.CacheModeWaitPingPool	PDB1	select /*+ first_rows(1) no_ex...
132.65	60	2.21	5.62	153.16	86.61	0.24	1t83y2u960dxd	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
115.65	3,735	0.03	4.90	178.12	64.93	0.44	c9171b67uapwk	OEM.PbsSystemPool	PDB1	BEGIN em_metric.bulk_insert_me...
76.80	5	15.36	3.25	82.35	93.25	0.00	7y0xytx71ss2g	MGMT_COLLECTION.COLLECTION_SUBSYSTEM	PDB1	CALL emd_rac.service_status_bu...
71.39	1,293	0.06	3.02	76.25	93.63	0.00	gk1qggpwcaugv	MGMT_COLLECTION.COLLECTION_SUBSYSTEM	PDB1	WITH CLS_INFO AS (SELECT DEST...
53.65	1	53.65	2.27	63.75	84.16	4.60	46z58hrjxrh4x	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
47.31	3,660	0.01	2.00	69.15	68.42	0.57	btf5nuff4cm3a	OEM.PbsSystemPool	PDB1	INSERT INTO EM_METRIC_VALUES (...
41.88	1	41.88	1.77	100.08	41.85	8.30	c23jwp2pztmbs	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
34.53	1	34.53	1.46	39.82	86.71	5.27	89byuzpjwmad4	DBMS_SCHEDULER	PDB1	/* MV_REFRESH (INS) */INSERT /...
34.45	138	0.25	1.46	39.33	87.58	0.07	4978nhr33jmp2	OEM.PbsDefaultPool	PDB1	begin EM_COMPLIANCE_LOADER.eva...
24.63	620	0.04	1.04	54.82	44.93	0.53	b12g3n0660rnc	OEM.PbsDefaultPool	PDB1	BEGIN EMD_LOADER.EXECUTE_POST_...
24.38	620	0.04	1.03	54.14	45.02	0.54	7x0yk3yrzv3rf	OEM.PbsDefaultPool	PDB1	BEGIN ECM_CT.POSTLOAD_CALLBACK...
23.64	28,577	0.00	1.00	35.84	65.98	0.05	9v2s6bjzpj9m0	OEM.PbsSystemPool	PDB1	UPDATE EM_METRIC_ITEMS SET IS_...

Back to SQL Statistics
Back to Top

SQL ordered by User I/O Wait Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- %Total - User I/O Time as a percentage of Total User I/O Wait time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 55.1% of Total User I/O Wait Time (s): 69
- Captured PL/SQL account for 39.3% of Total User I/O Wait Time (s): 69

User I/O Time (s)	Executions	UIO per Exec (s)	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id	SQL Module	PDB Name	SQL Text
22.92	179	0.13	33.27	49.22	20.54	46.58	bjdntqgd0fny4	OEM.PbsDeprecatedJobWorkerPool	PDB1	SELECT context_type AS name, t...
10.55	1	10.55	15.31	20.90	35.60	50.46	8h53rf67wxxh8	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
8.30	1	8.30	12.05	100.08	41.85	8.30	c23jwp2pztmbs	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
5.43	1	5.43	7.88	23.58	65.93	23.01	f8gwj7vn10q77	DBMS_SCHEDULER	PDB1	/* MV_REFRESH (DEL) */ delete ...
5.40	2	2.70	7.84	16.01	58.01	33.72	7dhg9ys7r45hx	Oracle Enterprise Manager.Metric Engine	PDB1	SELECT /*+ NO_STATEMENT_QUEUIN...
2.93	1	2.93	4.26	63.75	84.16	4.60	46z58hrjxrh4x	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
2.10	1	2.10	3.04	39.82	86.71	5.27	89byuzpjwmad4	DBMS_SCHEDULER	PDB1	/* MV_REFRESH (INS) */INSERT /...
0.86	1	0.86	1.24	96.36	19.11	0.89	ayjdzzm9ytydr	DBMS_SCHEDULER	PDB1	DECLARE job BINARY_INTEGER := ...
0.79	3,735	0.00	1.14	178.12	64.93	0.44	c9171b67uapwk	OEM.PbsSystemPool	PDB1	BEGIN em_metric.bulk_insert_me...
0.75	1	0.75	1.09	56.98	25.49	1.32	c3mqytm5242h1	DBMS_SCHEDULER	PDB1	/* MV_REFRESH (INS) */INSERT /...

PG_PROFILE

PG_PROFILE
Introduction

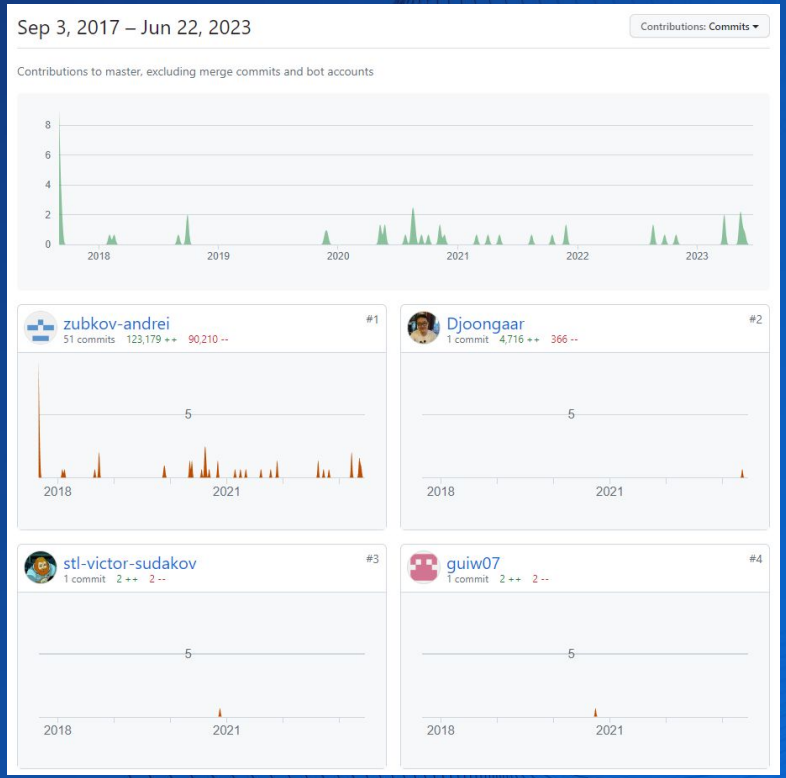
A good or the actual only(?) Example

Sample collection of

- System Catalogue, information_schema
- `PG_STAT_STATEMENTS`
- `PG_STAT_KCACHE`
- `PG_WAIT_SAMPLING`

PG_PROFILE
Development

Initiated by Andrei Zubkov
Individual Open Source license
https://github.com/zubkov-andrei/pg_profile
Starting Release v0.0.7 (Nov 2019)
Actual Release v4.2 (May 2023)
Pure PL/PGsql-based



PG_PROFILE

Example Report

Postgres profile report (58 -59)

pg_profile version 4.2

Server name: **node3_core15_postgres**

Report interval: **2023-06-25 19:00:01+00 - 2023-06-25 19:30:02+00**

Report sections

Server statistics

Database statistics

Database	Transactions			Block statistics			Tuples				
	Commits	Rollbacks	Deadlocks	Hit(%)	Read	Hit	Ret	Fet	Ins	Upd	D
postgres	1424			100	35	4964931	5180119	3097883	3878	3347	28
Total	1424			100	35	4964931	5180119	3097883	3878	3347	28

Session statistics by database

Database	Timings			Sessions			
	Total	Active	Idle	Established	Abondoned	Fatal	Killed
postgres	1497.34	3.49	0.12	495			
Total	1497.34	3.49	0.12	495			

Tablespace statistics

Tablespace	Path	Size	Growth
pg_default		90 MB	312 kB
pg_global		531 kB	

Wait sampling

Wait events types

Wait event type	Statements Waited (s)	%Total	All Waited (s)	%Total
Activity			7279.99	81.62
Client			1556.66	17.45
IO			0.39	
LWLock			0.03	
Timeout			82.59	0.93
Total			8919.66	100

Top wait events (statements)

Top wait events detected in statements execution

Wait event type	Wait event	Waited (s)	%Total

Top wait events (All)

Top wait events detected in all backends

Wait event type	Wait event	Waited (s)	%Total
Client	ClientRead	1556.66	17.45
Activity	AutoVacuumMain	1472.61	16.51
Activity	LogicalLauncherMain	1472.61	16.51
Activity	WalWriterMain	1472.5	16.51
Activity	BgWriterHibernate	1441.63	16.16
Activity	CheckpointerMain	1389.65	15.58
Timeout	CheckpointWriteDelay	82.55	0.93
Activity	BgWriterMain	30.99	0.35
IO	DataFileWrite	0.25	
IO	WALSync	0.09	
Timeout	VacuumDelay	0.04	
LWLock	WALInsert	0.03	
IO	DataFileFlush	0.02	
IO	DataFileSync	0.02	
IO	ControlFileSyncUpdate	0.01	

[284d83ba74]						
b663e72bbf906c1a [30ba98ac0c]	postgres	postgres	28		100	

Top SQL by shared blocks read

Query ID	Database	User	Reads	%Total	Hits(%)	Elapsed(s)	Rows	Ex

Top SQL by shared blocks dirtied

Query ID	Database	User	Dirtied	%Total	Hits(%)	Dirti
fb5ce79fc4bdf20f [0eeaf590a6]	postgres	postgres	832	832	100	90.
f8259241bf7cef37 [ab7afda8fa]	postgres	postgres	86	86	100	9.

Top SQL by shared blocks written

Query ID	Database	User	Written	%Total	%BackendW
fb5ce79fc4bdf20f [0eeaf590a6]	postgres	postgres	35	3.44	100

Top SQL by WAL size

Query ID	Database	User	WAL	%Total	Dirtied	WAL
fb5ce79fc4bdf20f [0eeaf590a6]	postgres	postgres	6227 kB	84.33	832	
f8259241bf7cef37 [ab7afda8fa]	postgres	postgres	115 kB	1.56	86	

rusage statistics

Top SQL by system and user time

Query ID	Database	User	User Time		System Time	
			Exec (s)	%Total	Exec (s)	%To
fb5ce79fc4bdf20f [0eeaf590a6]	postgres	postgres	1.42	76.83	0.06	42
4b097e9be01da20e [8093c244bf]	postgres	postgres	0.22	11.64	0.02	15
a75bdca7cac3636d [de1c8702bb]	postgres	postgres	0.07	4.05	0.01	10.75
cb3743bef713ee24 [12bdecef37]	postgres	postgres	0.04	1.91		2.96
f8259241bf7cef37 [ab7afda8fa]	postgres	postgres	0.02	1.28		

Complete list of SQL texts

Query ID	
8df9549c708fc694	SELECT DISTINCT ON (blocked_pid, locktype, blocking_pid) blocked_activity.pid AS blocked_pid, blocked_activ blocked_activity.query AS blocked_statement, blocking_activity.query AS current_statement_in_blocking_proces
8fbba6b1051e9899	SELECT ct.conname, ct.conkey, ct.confkey, nl.nspname AS fknsp, cl.relname AS fktab, nr.nspname AS refnsp, c
9c8db794731c91bc	SELECT nspname FROM pg_namespace nsp JOIN pg_proc pro ON pronamespace=nsp.oid WHERE proname = $1
9f4e11ca39a8b4c1	SELECT s.nspname AS schema_name, $1 AS package_name, f.proname AS function_name, $2::"char" AS function_type
a068feeeec1087c6	SET client_encoding TO 'utf8'
a1069251754491B9	SELECT nspname AS schema_name FROM pg_catalog.pg_namespace WHERE nspname = $1 OR nspname NOT LIKE $2
a75bdca7cac3636d	SELECT $2 AS package_name, f.proname AS function_name, $3::"char" AS function_type, f.prorettype::regtype A s.nspname = $1
ae247e8d0d89785f	SELECT i.schemaname AS schema_name, i.indexrelname AS index_name, s.idx_scan, s.idx_tup_read, s.idx_tup_fet
b663e72bbf906c1a	SELECT n.nspname AS schema_name, c.relname AS view_name, pg_relation_size(c.oid) / $1 AS mview_size_mb, pg_
ca069d9d4625d451	WITH pg_stat_statistics AS (SELECT p.queryid, p.dbid, p.userid, substring(p.query, $1, $2) AS short_query, LEFT JOIN pg_database d ON (p.dbid = d.oid) order by p.queryid desc
cb3743bef713ee24	SELECT pg_catalog.version() AS version_string, ((select setting from pg_settings where name = $1)::decimal AS effective_cache_size_mb, ((select setting from pg_settings where name = $13)::decimal * (select setting wal_buffers_mb, (SELECT pg_postmaster_start_time())::timestamptz AS server_start_time
d38f8743c701e8e6	SELECT COALESCE(sum($1), $2) AS number_of_prepared_transactions FROM pg_catalog.pg_prepared_xacts
d4e4e1cfd8c55bdd	SELECT n.nspname AS schema_name, c.relname AS table_name, pg_relation_size(c.oid) / $1 AS table_size_mb, pg_
db11d8257cef5f93	SELECT s.schemaname as schema_name, s.relname as table_name, s.seq_scan, s.seq_tup_read, s.idx_scan, s.idx_ s.relid = i.relid JOIN pg_class c ON s.relid = c.oid WHERE c.relkind != $1
dc6cee39214b6118	SELECT version()
f8259241bf7cef37	SELECT schemaname AS schema_name, tablename AS table_name, sml.relpages AS estimated_pages, target_pages, R $6 ELSE ((block_size - page_header) / estimated_bytes_per_tuple) END) AS target_pages, estimated_bytes_per_ ELSE null_bitmap_size % max_align END)) + item_ptr)::bigint AS estimated_bytes_per_tuple FROM (SELECT sche WHEN substring(v,$15,$16) IN ($17,$18,$19) THEN $20 ELSE $21 END AS tuple_header, CASE WHEN substring(v,$22
f915a933fb9cdb00	SELECT st.relid,st.schemaname,st.relname,st.seq_scan,st.seq_tup_read,st.idx_scan,st.idx_tup_fetch,st.n_tup_ins,st. relsize,$2 relsize_diff,class.reltablespace AS tablespaceid,class.reltoastrelid,class.relkind,class.relpage
fb28b5e485efe49b	SELECT st.*,stio.idx_blks_read,stio.idx_blks_hit,$1 relsize,$2,pg_class.reltablespace as tablespaceid,(ix. st.indexrelid) LEFT OUTER JOIN pg_catalog.pg_constraint con ON (con.conindid = ix.indexrelid AND con.contyp
fb5ce79fc4bdf20f	SELECT profile.take_sample()
45f64bd85b0f9e4	WITH max_age AS (SELECT $1 as max_old_xid, setting AS autovacuum_freeze_max_age FROM pg_catalog.pg_settings (oldest_current_xid/autovacuum_freeze_max_age::float)) AS percent_towards_emergency_autovac FROM per_databa
e42d4769165b4eb	SELECT setting FROM pg_settings WHERE name=$1
fed2778f1d749ae	SELECT $1
22ee9e843beld0bd	SELECT DISTINCT ON (name) name, setting, unit FROM pg_catalog.pg_settings
4b097e9be01da20e	SELECT c.relname AS view_name, c.relkind AS view_type, c.relispopulated AS ispopulated, spc.spcname AS table
4b5efda716747fde	SELECT datname AS database_name, datallowconn AS connections_allowed, pg_encoding_to_char(encoding) AS enco
4d65a4b5cf1d099d	SELECT datname AS database_name, pid AS procpid, usename, client_addr, client_port, backend_start, xact_sta
5a1c6d5a186940e9	SELECT f.funcid,f.schemaname,f.funcname,pg_get_function_arguments(f.funcid) AS funcargs,f.calls,f.total_tim
60ee2aded793ec13	SELECT c.relname AS index_name, r.relname AS table_name, i.indkey AS ind_keys FROM pg_catalog.pg_class c, p
62eebb8d9a4cba8d	SELECT checkpoints_timed, checkpoints_req, buffers_clean, buffers_checkpoint, maxwritten_clean, buffers_bac
6888a76611646404	SELECT c.relname AS table_name, array_length(array_agg(i.indexrelid), $2) > $3 AS has_primary_key FROM (SEL
6a2962ddfc52765f	SELECT n.nspname AS schema_name, c.relname AS index_name, pg_relation_size(c.oid) / $1 AS index_size_mb FRO

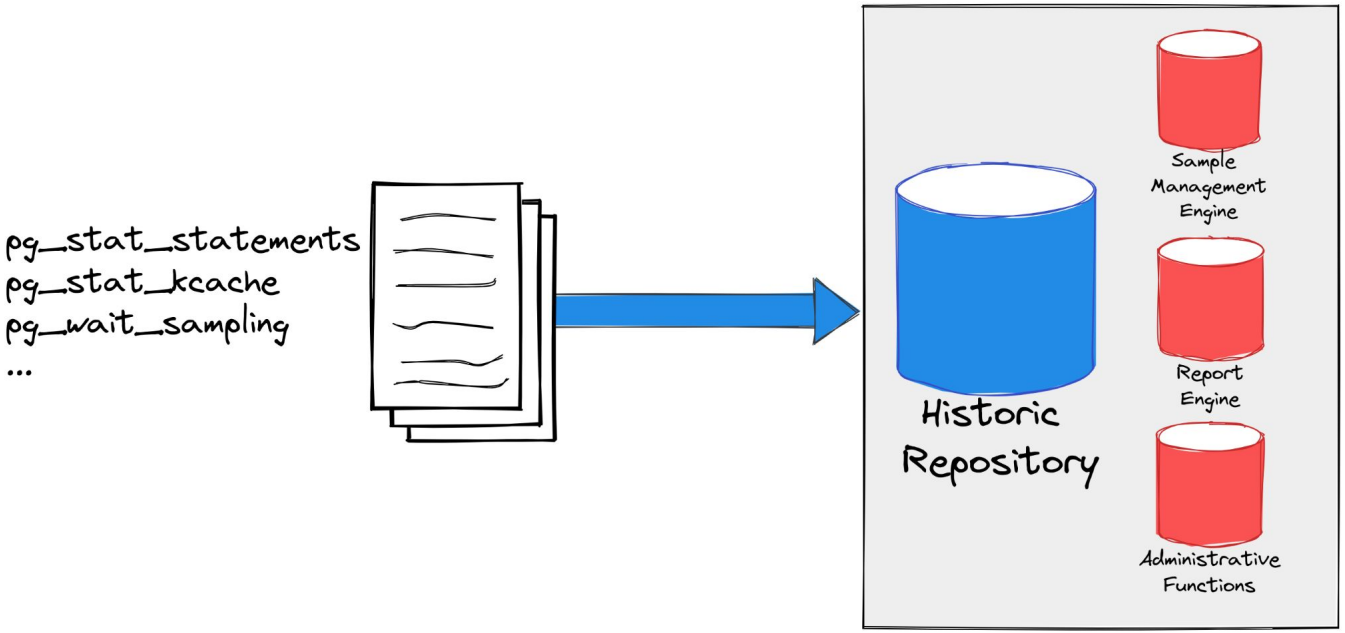
PG_PROFILE

What is the content of a Report?

Server Statistics
SQL query Statistics
Schema Object Statistics
User Function Statistics
Vacuum-related stats
Cluster settings

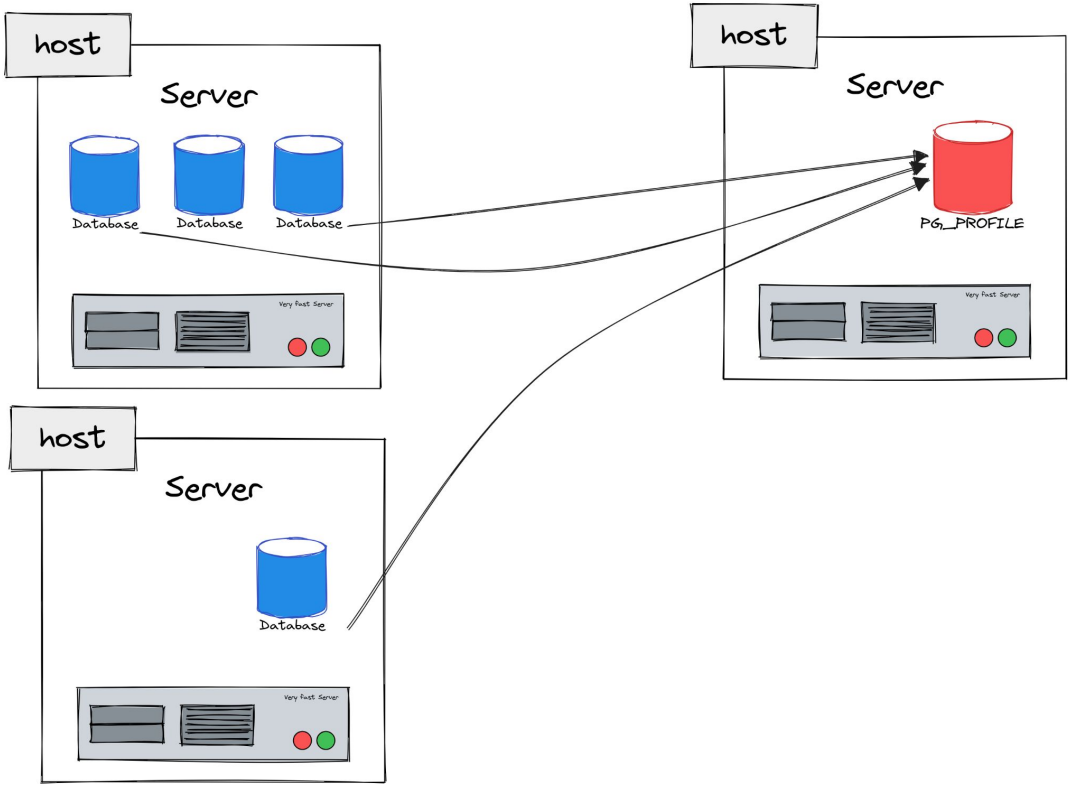
PG_PROFILE

Architecture



PG_PROFILE

Architecture



PG_PROFILE

Setup

Unfortunately not in any Repo yet (I guess)

But as usual an easy setup

```
# curl -LJO
https://github.com/zubkov-andrei/pg_profile/releases/download/4.2/pg_profile--4.2.tar.gz

# sudo tar xzf pg_profile--4.2.tar.gz --directory $(pg_config --sharedir)/extension
```

PG_PROFILE

Prerequisites on source DBs

Preload (wanted) Extensions

Set few recommended Parameters

```
# vi $PGDATA/postgresql.conf
…
shared_preload_libraries = 'pg_stat_statements, pg_wait_sampling, pg_stat_kcache'
…
track_activities = on
track_counts = on
track_io_timing = on
track_wal_io_timing = on
track_functions = all
```

PG_PROFILE
Prerequisites

Create Schema for Repository (optional)

```
CREATE SCHEMA profile;
```

Activate necessary Extensions

```
CREATE EXTENSION pg_profile SCHEMA profile;
CREATE EXTENSION dblink;
```

PG_PROFILE
Configuration

Consider extension parameters

```
pg_profile.topn = 20
pg_profile.max_sample_age = 7
pg_profile.track_sample_timings = off
pg_profile.max_query_length = 20000
```

As well for the related extensions, like e.g.

```
pg_stat_statements.max = 10000
pg_stat_statements.track = 'top'
```

EDB

PG_PROFILE

Adding Clusters/Servers for Collection

Add Server/Database

```
SELECT profile.create_server('core15','host=node0 dbname=postgres port=50150');
```

Other functions

```
profile.drop_server(server name)
profile.enable_server(server name)
profile.disable_server(server name)
profile.show_servers()
…
```

PG_PROFILE

Collecting Data

Take a sample

```
select * from profile.take_sample();
select * from profile.take_sample('core15');
```

Check existing samples

```
select * from profile.show_samples();
select * from profile.show_samples('core15');
```

PG_PROFILE

Collecting Data

Best Practice Strategy

- Frequented 30 Min Samples, starting point
- Consider manual created Samples
- Baselines

Putting into cron

```
*/30 * * * * psql -c 'SELECT profile.take_sample()' > /dev/null 2>&1
```

PG_PROFILE

Baselines

Tagged Group of Samples

Independent Retention

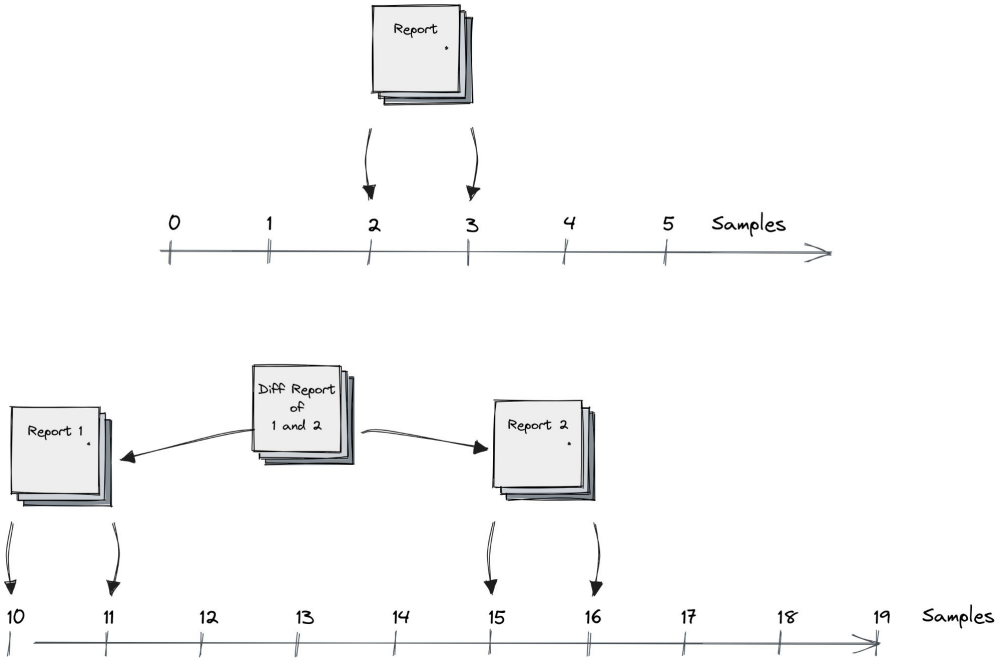
E.g. for bulk operations, load testings,...

Example handling

```
select * from profile.show_baselines();
select * from  profile.create_baseline('core15', 'pgbench_run' , 70, 71);
```

PG_PROFILE

Creating Reports



PG_PROFILE

Creating Reports

Standard Report

```
psql -Aqtc \
"SELECT profile.get_report('core15',8,9)" \
-o report_8_9.html
```

Diff Report Report

```
psql -Aqtc \
"SELECT profile.get_diffreport('core15', 8, 9, 11, 12)" \
-o diff_report_8_9-11_12.html
```

PG_PROFILE

A look into the repository schema

```
# \dt profile.*
                        List of relations
 Schema  |              Name               |       Type        |  Owner
---------+---------------------------------+-------------------+----------
 profile | baselines                       | table             | postgres
 profile | bl_samples                      | table             | postgres
 profile | funcs_list                      | table             | postgres
 profile | import_queries                  | table             | postgres
 profile | import_queries_version_order    | table             | postgres
 profile | indexes_list                    | table             | postgres
 profile | last_stat_archiver              | table             | postgres
 profile | last_stat_cluster               | table             | postgres
 profile | last_stat_database              | partitioned table | postgres
 profile | last_stat_database_srv1         | table             | postgres
 profile | last_stat_database_srv2         | table             | postgres
 profile | last_stat_database_srv4         | table             | postgres
 profile | last_stat_indexes               | partitioned table | postgres
 profile | last_stat_indexes_srv1          | table             | postgres
 profile | last_stat_indexes_srv2          | table             | postgres
 profile | last_stat_indexes_srv4          | table             | postgres
 profile | last_stat_kcache                | partitioned table | postgres
 profile | last_stat_kcache_srv1           | table             | postgres
 profile | last_stat_kcache_srv2           | table             | postgres
 profile | last_stat_kcache_srv4           | table             | postgres
 profile | last_stat_statements            | partitioned table | postgres
 profile | last_stat_statements_srv1       | table             | postgres
 profile | last_stat_statements_srv2       | table             | postgres
 profile | last_stat_statements_srv4       | table             | postgres
 profile | last_stat_tables                | partitioned table | postgres
 profile | last_stat_tables_srv1           | table             | postgres
 profile | last_stat_tables_srv2           | table             | postgres
 profile | last_stat_tables_srv4           | table             | postgres
 profile | last_stat_tablespaces           | partitioned table | postgres
 profile | last_stat_tablespaces_srv1      | table             | postgres
 profile | last_stat_tablespaces_srv2      | table             | postgres
 profile | last_stat_tablespaces_srv4      | table             | postgres
…
```

```
…
profile | last_stat_user_functions      | partitioned table | postgres
 profile | last_stat_user_functions_srv1 | table             | postgres
 profile | last_stat_user_functions_srv2 | table             | postgres
 profile | last_stat_user_functions_srv4 | table             | postgres
 profile | last_stat_wal                 | table             | postgres
 profile | report                        | table             | postgres
 profile | report_static                 | table             | postgres
 profile | report_struct                 | table             | postgres
 profile | roles_list                    | table             | postgres
 profile | sample_kcache                 | table             | postgres
 profile | sample_kcache_total           | table             | postgres
 profile | sample_settings               | table             | postgres
 profile | sample_stat_archiver          | table             | postgres
 profile | sample_stat_cluster           | table             | postgres
 profile | sample_stat_database          | table             | postgres
 profile | sample_stat_indexes           | table             | postgres
 profile | sample_stat_indexes_total     | table             | postgres
 profile | sample_stat_tables            | table             | postgres
 profile | sample_stat_tables_total      | table             | postgres
 profile | sample_stat_tablespaces       | table             | postgres
 profile | sample_stat_user_func_total   | table             | postgres
 profile | sample_stat_user_functions    | table             | postgres
 profile | sample_stat_wal               | table             | postgres
 profile | sample_statements             | table             | postgres
 profile | sample_statements_total       | table             | postgres
 profile | sample_timings                | table             | postgres
 profile | samples                       | table             | postgres
 profile | servers                       | table             | postgres
 profile | stmt_list                     | table             | postgres
 profile | tables_list                   | table             | postgres
 profile | tablespaces_list              | table             | postgres
 profile | wait_sampling_total           | table             | postgres
(64 rows)
```

PG_PROFILE

A look into the repository schema

Global Retention Policy

```
pg_profile.max_sample_age
```

Server Retention Policy

```
pg_profile.set_server_max_sample_age()
```

PG_PROFILE

A look into the repository schema

Data Growth?

```
WITH schemas AS (
SELECT schemaname as name, sum(pg_relation_size(quote_ident(schemaname) || '.' ||
quote_ident(tablename)))::bigint as size
FROM pg_tables GROUP BY schemaname ),db AS ( SELECT pg_database_size(current_database()) AS size
) SELECT schemas.name,
       pg_size_pretty(schemas.size) as absolute_size,
       schemas.size::float / (SELECT size FROM db)  * 100 as relative_size
FROM schemas;
```

```
        name         | absolute_size |    relative_size
---------------------+---------------+---------------------
 public              | 41.00 MB      |  51.55
 pg_catalog          |  0.50 MB      |   6.02
 information_schema  |  0.09 MB      |   0.10
 profile             | 18.00 MB      |  22.04

(4 rows)
```

Conclusion and Final Words

Conclusion and Final Words

My 2 Cents

Very clean, pragmatic way to handle Performance data

Sustainable repository approach

Handy reports, also for developers

Big Point for considering Oracle folks

Conclusion and Final Words

What is missing?

Dream of having such in contrib (including `pg_wait_sampling`) :-)

…or at least in all Repos

Own Job handling would be nice, but cron is ok

Much room for even more improvements

Conclusion and Final Words

Coming soon

Work in progress:

Blog/Talk about “Post-Migration Sizing for PostgreSQL based on Oracle AWR”

See you (hopefully) at…

PGDAY UK	2023.pgday.uk	(London, Sep 12)
PGCONF NYC 2023	2023.pgconf.nyc	(New York City, Oct 3-5)
POUG 2023	poug.org/en	(Warsaw, Nov 17-18)
DOAG K&A 2023	anwenderkonferenz.doag.org	(Nuremberg, Nov 21-24)
PGConf.EU 2023	2023.pgconf.eu	(Prague, Dec 12-15)

EDB

THANK YOU!